Docket No.: M&N-IT-465

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Chee Hong Liau

Filed : Concurrently herewith

Title : Method of Processing Test Patterns for an Integrated Circuit

CLAIM FOR PRIORITY

Hon. Commissioner for Patents,

S i r :

Claim is hereby made for a right of priority under Title 35, U.S. Code, Section 119, based upon European Patent Application No. 02090272.2, filed July 19, 2002.

A certified copy of the above-mentioned foreign patent application is being submitted herewith.

Respectfully submitted,

LAURENCE A. GREENBERG
REG. NO. 29,308

For Applicant

Date: July 18, 2003

Lerner and Greenberg, P.A.
Post Office Box 2480
Hollywood, FL 33022-2480
Tel: (954) 925-1100
Fax: (954) 925-1101
/bb

Europäisches Patentamt | European Patent Office | Office européen des brevets

Blatt 2 der Bescheinigung
Sheet 2 of the certificate
Page 2 de l'attestation

Anmeldung Nr.:
Application no.: 02090272.2
Demande n°:

Anmeldetag:
Date of filing: 19/07/02
Date de dépôt:

Anmelder:
Applicant(s):
Demandeur(s):
Infineon Technologies AG
81669 München
GERMANY

Bezeichnung der Erfindung:
Title of the invention:
Titre de l'invention:
Method of processing test patterns for an integrated circuit

In Anspruch genommene Priorität(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)

Staat: State: Pays:	Tag: Date: Date:	Aktenzeichen: File no. Numéro de dépôt:

Internationale Patentklassifikation:
International Patent classification:
Classification internationale des brevets:
G01R31/3183

Am Anmeldetag benannte Vertragstaaten:
Contracting states designated at date of filing: AT/BG/BE/CH/CY/CZ/DE/DK/EE/ES/FI/FR/GB/GR/IE/IT/LI/LU/MC/NL/
Etats contractants désignés lors du depôt:

Bemerkungen:
Remarks:
Remarques:

Europäisches Patentamt
European Patent Offic
Office eur péen des brev ts

Bescheinigung Certificate Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

02090272.2

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

EPO-BERLIN
19 -07- 2002

Description

Method of processing test patterns for an integrated circuit

To achieve high perfomance and high integration density, the dimensions of integrated circuit components are scaled down more and more. In particular, transistor dimensions are scaled down while lower power dissipation is achieved by scaling down the supply voltage. However, due to high packing density of transistors, the power supply current is increasing, and hence, large current swings within a short period of time can cause considerable noise. As a consequence, one difficulty circuit designers face is the power delivery of very high performance circuits due to the severe switching noise.

In order to verify the function of a newly designed integrated circuit, the circuit is first simulated and then tested. During simulation, multiple input signals are applied to the inputs of the circuit, and the output signals of the circuit calculated. The input signals are referred to as test patterns. If the output signals do not sufficiently approximate preset target signals, the circuit is redesigned and resimulated.

Subsequently, when simulation is completed, a chip containing the integrated circuit is manufactured and tested using ATE (Automatic Test Equipment). The ATE also applies a test pattern to the circuit. The test pattern for the ATE has to be input manually by a user. Generally, the same test pattern that has been used for simulation is also used for testing. If the output signals generated by the circuit in response to the test pattern of the ATE deviate from preset target signals, the circuit is redesigned, resimulated and retested.

As the complexity of integrated circuits increases,

integration density and functionality increases dramatically. The simultaneous switching of a large number of transistors induces a large current spike. The switching noise on the power distribution network must be suppressed to a tolerable level to ensure the reliability of the circuit. In order to efficiently combat the switching noise, estimation of the worst case switching noise is required.

On way of determining the worst case switching noise is to simulate all combinations of input patterns to determine which combination will induce the maximum switching noise. However, the complexity of the solution space is exponentially proportional to the number of primary inputs of the system. Accordingly, it would require an enormous time to process the entire solution space for even a moderately complex system.

As a consequence, it is almost impossible to determine all test patterns that cause worst case switching noise by simalation or testing. Accordingly, a small set of test patterns that cause at least some worst case scenarios can be selected. However, this way, the simulation or testing of the integrated circuit may not be satisfactory.

The designer thus has to accept either enormous time requirements for simulation and testing, or potentially insufficient simulation or testing efficiency. This is clearly undesirable.

To this end, some approaches have been proposed to deal with these problems. In "Estimation of Switching Noise on Power Supply Lines in Deep Sub-micron CMOS circuits", Shiyou Zhao and Kaushik Roy, 13th International Conference on VLSI Design, IEEE January 2000, there is proposed a probabalistic approach to determine the lower bound of the worst case switching noise on power supply lines. The algorithm described therein traces the worst case input patterns which

induces the steepest maximum switching current spike and therefore the maximum switching noise. This is based on the observation that the maximum switching noise is directly related to the steepest maximum switching current spike.

In this approach, the design of an integrated circuit is simulated by applying randomly generated input signal vectors to the inputs of the circuit. For each input vector pair, the simulated peak switching current is determined. The worst case input vector pairs feed, as initial population, a genetic algorithm. The genetic algorithm is designed to single out the near optimal input pattern(s) that induce the steepest maximum switching current spike and, therefore, the worst case switching noise. The worst case input patterns are then used in HSPICE simulation of the circuits to extract the exact current waveform.

One problem associated with this approach is the difficulty of generating suitable random test patterns. The larger the number of random test patterns, the higher the likelihood of generating a test pattern which approximates the worst case sufficiently. However, since the simulation of each test pattern is time consuming, the simulation of a large number number of test patterns is not practical.

In particular, if a genetic algorithm is used, it is too time consuming to simulate every single random pattern out of every new pattern population before the algorithm is able to determine which of the patterns of the population is to be selected for further optimization. Therefore, this method becomes saturated by the number of trial random patterns in each pattern population. It is suitable for small circuits. However, it could take up to years to perform a full chip simulation of a large circuit using even the fastest simulation applications.

The present invention aims to address these problems.

According to one aspect of the invention, there is provided a method of approximating the behaviour of an integrated circuit, the method comprising:

(a) applying a set of test patterns to a system for testing or simulating an integrated circuit;

(b) applying the set of test patterns to a neural network;

(c) comparing the outputs of the system for testing or simulating the integrated circuit and the outputs of the neural network; and

(d) adapting parameters of the neural network to approximate the behaviour of the integrated circuit on the basis of the comparison.

In particular, the system for testing or simulating the integrated circuit may be an automatic test equipment (ATE), and the set of test patterns is applied to the integrated circuit via the automatic test equipment. The neural network may be integrated in the ATE. It may be implemented using any known ATE.

The invention is based on the idea that the dynamic behaviour of the integrated circuit device can be learnt from a set of random test patterns using a neural network.

This mode of operation of the neural network can be referred to as leanring mode.

After the learning process has been completed, the ATE is able to perform test pattern classification. The ATE may thus select sub-optimal patterns for subsequent simulation or testing of the integrated circuit. The selected test patterns sufficiently approximate worst case scenarios.

Accordingly, the invention also provides for a method of selecting test patterns, the method comprising:

(m) applying a test pattern to a neural network whose parameters have been adapted to approximate the behaviour of an integrated circuit according to the above described method;

(n) processing the output of the neural network to determine whether predetermined criteria are met; and

(o) selecting for storage the test pattern if the predetermined criteria are met.

Using this method, a sufficient number of test patterns are provided for an efficient simulation or testing of the integrated circuit.

This mode of operation of the neural network can be referred to as operation mode.

Accordingly, there is also provided a method of testing an integrated circuit; the method comprising: applying test patterns that have been selected according to the above described method of selecting test patterns to the integrated circuit using automatic test equipment (ATE).

In addition, there is provided a method of simulating an integrated circuit; the method comprising: applying test patterns that have been selected according to the above described method of selecting test patterns to a simulator for simulating an integrated circuit.

According to a preferred embodiment of the invention, the test patterns that have been selected in accordance with the above described neural network-based approach are further optimized using a genetic algorithm.

Thus, there is provided a method of providing a test pattern for the simulation and/or test of the layout of an integrated circuit, the method comprising the steps of:

(A) providing a set of test patterns consisting of test patterns selected in accordance with the above described method of selecting test patterns;

(B) applying the set of test patterns to the integrated circuit using automatic test equipment (ATE);

(C) determining the outputs of the integrated circuit;

(D) processing the outputs to determine whether predetermined test criteria are met;

(E) depending on the determination in step (D), generating a new set of test patterns on the basis of the set of test patterns provided by step (A) using a genetic algorithm.

This combination of neural network- and genetic algorithm-based approaches further increases the chances of finding test patterns that sufficiently approximate worst case scenarios of operation of the integrated circuit.

The method employs a genetic algorithm (optimization method) to optimize a set of pre-selected patterns based on measurements using an ATE. Thereby, a set of worst case noise patterns can be selected automatically. By using ATE for the processing of the test patterns, performance is greatly enhanced compared to approaches based on simulation.

Test patterns generated accordingly can additionally be re-simulated for further detail design analysis and improvement.

The genetic algorithm approach can be implemented using existing ATEs.

The invention will now be described, by way of example only, with reference to the accompanying drawings, of which:

Figure 1 illustrates schematically a classification of three groups of test patterns;

Figure 2 illustrates a process representing the neural network learning mode according to an embodiment of the invention;

Figure 3 illustrates a process representing the neural network operation mode according to an embodiment of the invention;

Figure 4 illustrates a schematic flow diagram of a genetic algorithm using intital random pattern generation; and

Figure 5 illustrates a schematic flow diagramm of a method combining neural network-based pre-selection of test patterns and subsequent optimization using a genetic algorithm.

In accordance with one embodiment of the invention, a neural network model is implemented into the ATE. That is, the ATE is able to learn automatically from test runs of an integrated circuit performed on the ATE. After training of the neural network has been completed, it is able to identify which group of test patterns belong to a sub-optimal set. For this purpose, some million test patterns are selected via the neural network. Subsequently, the pre-selected sub-optimal pattern is simulated (simulation approach) or measured (ATE approach) to further determine which of the patterns fulfill predetermined criteria. Those patterns which fulfill the criteria are selected for storage.

Figure 1 illustrates schematically a classification of three groups of patterns. Since the simulation approach- or ATE approach-based selection of patterns is performed on the basis of a sub-optimal set of patterns (group C), the speed and efficiency compared to conventional methods for identifying suitable test patterns is considerably improved. This approach is referred to as maximum approximation.

Suppose, as illustrated in Figure 1, there are $P_N \in \{P_N^A \ P_N^B \ P_N^C\}$, $N > 0$, three groups of test patterns within a full range of all possible test patterns. Instead of searching through all possible patterns, the neural network may learn and distinguish between different groups of test patterns using an ATE-based training program. Accordingly, the test patterns are pre-classified.

In the maximum approximation algorithm mode, the sub-optimal set P_N^C and $P_N^B \cap P_N^C$ are generated based on neural network decisions, and a sub-optimal pattern set forms a new pattern population.

This process can be repeated iteratively on the basis of the new pattern population, thereby to select the best group of patterns out of the sub-optimal population, and so on.

A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on that of the animal neuron. The network ability of the network is stored in the inter-unit connection strenghts, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns.

Figure 2 illustrates a process representing the neural network learning mode according to an embodiment of the invention. The neural network is based on back-propagation net characteristics to perform a pattern classification task. In the beginning, the neural network learns from a set of random patterns. The test results are supervised by a test system (ATE). The learning process is terminated when the learning error is less than a predetermined value. Subsequently, a neural network learning weight ("brain") file is generated. This file is then used in operation mode to perform a pattern classification task.

Figure 3 illustrates a process representing the neural network operation mode according to an embodiment of the invention. In the operation mode, the neural network is able to perform pattern classification based on previous learning experience (contained in the NN brain file) for pattern approximation and selection. The procedure of pattern selection may be based on a very small set of NN pattern populations. For example, one NN pattern population may include six NN decision patterns. The neural network first determines whether any pattern out of these six NN decision patterns belong to a potential maximum current group (sub-optimal group). If yes, then this pattern is selected. If no, then the search is repeated using the same procedure. In the final network classification, only those patterns are selected which cause a higher dynamic current than patterns that have been tested in real measurements (RSMA). For example, using this approach, 6*100 = 600 patterns can be classified, of which only 100 patterns require testing through measurement to determine if they cause higher dynamic currents, while the other 500 patterns are classified by the neural network.

An implementation of the neural network pattern learning process and the neural network pattern classification process is given in Annex 1 and Annex 2, respectively.

In one embodiment of the invention, the neural network is a back-propagation neural network. Back-propagation is a supervised learning algorithm mainly used by multi-layer-perceptrons in order to change the weights associated with the net's hidden neuron layer(s).

Another embodiment of the invention will now be described by reference to Figures 4 and 5. In this embodiment, test patterns selected by the neural network are further optimized using a genetic algorithm.

For the purposes of illustration, Figure 4 shows a schematic flow diagram of a genetic algorithm without pre-selection of test patterns by a neural network (instead, the initial patterns are generated on a random basis).

Figure 5 illustrates a schematic flow diagramm of a method combining neural network-based pre-selection of test patterns and subsequent optimization using a genetic algorithm, in accordance with an embodiment of the invention.

Genetic algorithms are based on the principles of natural selection. In particular, genetic algorithms are stochastic search methods which simulate natural biological evolution. The algorithms operate on the basis of a population of potential solutions and, applying the principle of "survival of the fittest" to these potential solutions, produce a better approximation of a target solution in each iteration of the algorithm.

Each iteration of the algorithm produces a new generation of approximations. The approximations of each generations are created by the process of selecting individuals according to their level of "fitness" in the problem domain. The selected individuals are bred with one another using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited for their environment than the individuals from which they were created, just as in natural adaptation.

Accordingly, genetic algorithms model natural processes such as selection, cross over, recombination and mutation.

Figure 4 shows a method for detecting the worst case current consumption/peak current pattern (RSMA) based on a genetic algorithm. This method operates on the basis of populations of individual patterns instead of a single pattern solution. In this way, the search for better approximations can be

performed in a parallel manner. Therefore, this method is more efficient than single pattern searching processes using dynamic random algorithm methods.

Genetic algorithms may be employed for the simulation of an integrated circuit design in order to solve the worst case pattern search problem. The efficiency of genetic searching procedures is largely dependent on the number of pattern populations and the number of test patterns in each pattern population. However, as indicated above, the simulation-based approach forms a limitation if genetic algorithms are to be employed. The genetic selection procedure has to evaluate every "fitness" (dynamic peak/averaged current) of the test patterns in each pattern population. For example, there may be 200 pattern populations each including 20 patterns. Thus, the genetic algorithm has to evaluate the fitness of 200 * 20 = 4000 patterns. If each test pattern is a 50 cycles test pattern which requires 30 minutes of simulation time (e.g. EPIC oder SPICE simulator), then the total required searching and simulation time is 4000 * 30 minutes = 120000 minutes, i.e. approximately 83 days of non stop simulation in order to process 200 pattern populations only.

In addition, the full pattern combination domain increases proportionally to the complexity of VLSI or ULSI designs. Therefore, a subset of 200 pattern populations is only a very small subset of the full pattern combination domain.

In contrast, when using a genetic algorithm together with ATE, many more pattern populations per time unit can be processed. This is because the testing of an integrated circuit using ATE is considerably faster than simulation using conventional systems. Accordingly, the approximation of worst case test patterns in a given period of time is much more accurate.

An implementation of a dynamic genetic algorithm for use with ATE is presented in the following. At the beginning of the computation, a number of individual random patterns

$$P_N^{POP} = (p_1, p_2, \ldots, p_N)$$

are randomly generated and initialized, wherein N is the maximum number of random patterns, and POP is the maximum number of pattern populations.

Subsequently, for each individual pattern $(p_1, p_2, \ldots, p_N)$, the objective functions

$$I_{peak}(\forall\ I_{sample}(P_N, SRMS))$$

and

$$I_{averaged}(P_N, SRMS)$$

are evaluated using equation (1):

$$I_{Measurement}(P_N,T) = \frac{V_{DD}(P_N,T)}{R_{eff}} + \frac{1}{L_{eff}}\int_{T_{min}}^{T_{max}} V_{DD}(P_N,T)dT + \Delta I_{CMOS}(P_N,T),\ \forall T, P_N > 0$$
$$T = SRMS(T_{min},T_{max}) \Rightarrow Random_Float_Number(T_{min},T_{max})$$
$$T_{max} \geq T_{min},\ \forall T_{min},T_{max} > 0,\ I_{Measurement}(P_N,T) \in \{I_{peak}\quad I_{averaged}\}$$

The first (initial) generation is thus produced, and the averaged fixness of the individual patterns $(p_1, p_2, \ldots, p_N)$ is calculated using equation (2):

$$Averaged_Fixness(Fixness(P_N^{POP})) = \frac{\sum_{N=0}^{N} I_{Measurement}(P_N)}{N},\ N, P_N > 0$$
$$Fixness(P_N) = I_{Measurement}(P_N,T) \in \{I_{peak}\quad I_{averaged}\}$$

If the optimization criteria

$$(\text{Averaged_Fixness}(I_{\text{Measurement}}(P_N^{POP})) < I_{MAX_REF})$$

is not met for any existing population, a new population is created on the basis of the existing population. Individual patterns are selected according to their fitness for the production of offspring (loop1 in Figure 4).

In this selection approach, the basic concept of tournament selection is employed. That is, only the best individual pattern from the existing population is selected as a parent. This process is repeated until a pre-defined percentage of best patterns has been selected:

(3)

$$Sorting(I_{Measurement}(P_N) \in \{I_{\min}(P_{N_{\min}}) \quad \ldots\ldots \quad I_{\max}(P_{N_{\max}})\}) \Rightarrow Parent(I_{Measurement}(P_N))$$
$$N \in \{N_{\min} \quad N_{\max}\} \qquad\qquad N \in \{N_{\min} = N_{\max} - (N_{\max} \times B) \quad N_{\max}$$

wherein B is the pre-defined percentage of the best pattern group. The sorting function first re-arranges the test patterns from minimum to maximum according to their fixness values. Subsequently, the parent selection is generated in random sequence based on the new sub-optimal fixness range N, which is calculated using B. Parents (selected patterns) are combined using cross over (4), re-combined (5) and mutated (6) in order to produce offspring:

$$\begin{aligned} CrossOver(P_N(C_1,C_2),P_{N+1}(C_3,C_4)) &\Rightarrow Upper_CrossOver(P_N(C_3,C_2),P_{N+1}(C_1,C_4)) \\ &\Rightarrow Lower_CrossOver(P_N(C_1,C_4),P_{N+1}(C_3,C_2)) \\ &\Rightarrow Stripe_CrossOver(P_N(C_4,C_3),P_{N+1}(C_2,C_1)) \end{aligned} \quad (4)$$

where C is the test pattern content which is selected for cross over of two patterns. In the cross over process, upper, lower or stripe cross over methods are performed in random sequence, and the contents of two cross over patterns are exchanged in order to produce two new offspring patterns.

Thereafter, the re-combination equation (5) is used to select the best fixness pattern out of two new cross over offspring patterns:

$$Recombination(P_N, P_{N+1}) \Rightarrow I_{maximum}(P_N, P_{N+1}) \Rightarrow I_{Best}(P_M),\ N, M, P_N > 0 \qquad (5)$$

$$Mutation(P_M(C_1, C_2, C_3, C_4 \ldots C_y)) \Rightarrow P_M(C_1 + R_1, C_2 + R_2, \ldots\ldots C_y + R_y)$$
$$R_{y_s} \in \{1 \quad 0 \quad -1\},\ M, P_M, y > 0 \qquad (6)$$

where M is the number of new selected offspring patterns to form the new population. After recombination, the offspring undergoes mutation. Offspring variables are mutated by the addition of small random values

$$Ry \in \{1 \quad 0 \quad -1\}.$$

The mutation process helps to improve the optimization search process.

Finally, all offspring patterns are inserted into the population, replacing the parents (original pattern population) and producing a new generation. This cycle (loop 1 in Figure 4) is performed until the optimiziation criteria are met.

If the fitness does not improve after a pre-defined number of genetic breeding generations, a new pattern population (loop 2 in Figure 4) will be generated in random sequence. This combination greatly increases the chances of finding worst case test patterns.

A complete implementation of this algorithm using ATE J973 is given in Annex 3.

Figure 5 illustrates a flow diagramm of a method combining neural network-based pre-selection of test patterns and subsequent optimization using a genetic algorithm.

A complete implementation of this combined approch using ATE J973 is given in Annex 4.

It is to be noted that the invention is not restricted to the embodiments and implementations described herein but encompasses modifications and variations within the scope of the invention as determined from the claims.

Annex 1: Neural Network Pattern Learning Implementation Using ATE J973

Start Neural Network Training Using ATE: Circuit Initialization

Default AC/DC Specification Initialization.
DP Dummy Pattern : Vector Memory Initialization

INPUT: $\{N \quad Vector_cycle \quad DP \quad Auto_Range \quad \varepsilon \quad G \quad Epoch \quad Max_Loop \quad EX \quad File_Name$

Check if Input valid?
else Input Error! exit(1).

For $P1 = 0, 1, 2, 3, \ldots\ldots\ldots, Auto_Range{+}1$ do : **(Automatic Input & Output Range Calculation)**
{

$Random_Pattern_Generation \Rightarrow P \in (p_1, p_2, \ldots., p_N)$: Random Pattern Population
$\forall vector_cycle, N > 0$

$X(P_N, I_{peak/averaged}(P_N)) \in \{X_1 \quad X_2 \quad \ldots\ldots\ldots \quad X_i\}, \; X_{offset} = (C_{\max} - C_{\min}) \times R_{offset}, \; R_{offset}, i > 0$
$X_1 \in \{X_{\min} = X_{\min} + X_{offset}\}$
$X_2 \in \{X_{mid11} = X_{\min} - 1 \quad X_{mid12} = X_{\min} + X_{offset}\}$
$X_3 \in \{X_{mid21} = X_{mid12} - 1 \quad X_{mid22} = X_{mid21} + X_{offset}\}$
$X_4 \in \{X_{\max} = X_{mid22} - 1\}$
}

Neural Network Training Loop:
{

$Random_Pattern_Generation \Rightarrow P \in (p_1, p_2, \ldots., p_N)$: Random Pattern Population
$\forall vector_cycle, N > 0$

$$Vector_Code_Matrix(P_N(Vector_Cycles)) \Rightarrow \begin{bmatrix} P_0(Vector_Cycles) \\ P_1(Vector_Cycles) \\ \vdots \\ P_N(Vector_Cycles) \end{bmatrix}$$

$P_N(Vector_Cycles) \in P_N(vector_encode(\forall signal_bus), Vector_Cycles))$ **(ATE Training Set)**

$Pattern_Generator(Vector_Memory(P_N))$
$\Rightarrow Pattern_Controller(Vector_Memory(P_N)) \quad N > 0$ *(Pattern Executor)*

Start Pattern Generator: $P_N(T_{\min}, T_{\max}) \Rightarrow Dynamic_Pattern$

Start Current Measurement & Calculation:

$$I_{Measurement}(P_N,T)=\frac{V_{DD}(P_N,T)}{R_{eff}}+\frac{1}{L_{eff}}\int_{T_{min}}^{T_{max}}V_{DD}(P_N,T)dT+\Delta I_{CMOS}(P_N,T),\ \forall T,P_N>0$$

$$T=SRMS(T_{min},T_{max})\Rightarrow Random_Float_Number(T_{min},T_{max})$$

$$T_{max}\geq T_{min},\ \forall T_{min},T_{max}>0$$

Stop Pattern Generator: $P_N : I_{peak}(\forall I_{sample}(P_N,SRMS)),\ I_{averaged}(P_N,SRMS)$

$$Z^k=I_{Measurement}(P_N,T)\in\{I_{peak}\quad I_{averaged}\}$$

$X^k(P_N,I_{Measurement}(P_N))\in\{X_1^k\quad X_2^k\quad .....\quad X_i^k\}$ **(Neural Network Learning Set)**

$$X_j=\sum_i W_{ji}a_i$$

$$Y(X)=\frac{1}{1+e^{-X\times G}}$$

$$E=\frac{1}{2}\sum_{k=1}^{s}\sum_{j=1}^{n}(Y_j^k-Z_j^k)^2$$ **(Supervising learning via ATE)**

$$\frac{\partial E}{\partial w_{ji}}=\sum_k(Y_j^k-Z_j^k)\times Y_j^k(1-Y_j^k)\times X_i^k$$

$$w_{ji}=w_{ji}-\varepsilon\times(\frac{\partial E}{\partial w_{ij}})\times G$$

$E(P_N)+=E(P_N)$ (pattem population Learning Error calculation)

$Training_Loop++$

if (Tranining _ Loop > Epoch) (pattern population learning done)

{ $E=\frac{\sum E(P_N)}{Epoch}$ }

if (Tranining _ Loop > Max _ Loop)

{ So Far Neural Network Learning File (*File_Name*) Generation, exit(1) }

if (E > EX)

{ Go To **Neural Network Training Loop : Learn Again!!** Initialize: $E=0$ }

else

{ Expected Neural Network Learning File (*File_Name*) Generation, exit(1) }

} End Of Neural Network Learning

Final Neural Network Learning Plot Generation (Figure 27).

End Of Neural Network Learning via ATE

Annex 2: Neural Network Pattern Classification Implementation Using ATE J973

Start NN_MA: Circuit Initialization

Default AC/DC Specification Initialization.
DP Dummy Pattern : Vector Memory Initialization

INPUT: $\{N \quad Vector_cycle \quad DP \quad Max_Loop \quad File_Name\}$

Check if Input valid?
else Input Error! exit(1).

$NN_Learning_File(File_Name) \in \{w_{ij} \quad \varepsilon \quad G \quad X^k\}$

NN_MA_Loop:
{

$Random_Pattern_Generation \Rightarrow P \in (p_1, p_2, ..., p_N)$: Random Pattern Population

$\forall vector_cycle, N > 0$

$X^k(P_N, I_{Measurement}(P_N)) \in \{X_1^k \quad X_2^k \quad ..... \quad X_i^k\}$ **(Neural Network Evaluation Set)**

$$X_j = \sum_i W_{ji} a_i$$

$$Y_N(X) = \frac{1}{1 + e^{-X \times G}}$$

$if\ (\forall Y_N(X) < Y_{sub_optimal_set})$
{ Go to **NN_MA_Loop : New Pattern Population Generation !!!** }
else
{ $Sorting(P_N, Y_N(X)) \Rightarrow P_M$ (sub-optimal set based on **neural netw rk)** }

$$Vector_Code_Matrix(P_M(Vector_Cycles)) \Rightarrow \begin{bmatrix} P_0(Vector_Cycles) \\ P_1(Vector_Cycles) \\ \vdots \\ P_M(Vector_Cycles) \end{bmatrix}$$

$P_M(Vector_Cycles) \in P_M(vector_encode(\forall signal_bus), Vector_Cycles))$

$Pattern_Generator(Vector_Memory(P_M))$
$\Rightarrow Pattern_Controller(Vector_Memory(P_M))\ \ M > 0$ *(Pattern Executor)*

Start Pattern Generator: $P_M(T_{min}, T_{max}) \Rightarrow Dynamic_Pattern$

Start Current Measurement & Calculation:

$$I_{Measurement}(P_M,T) = \frac{V_{DD}(P_M,T)}{R_{eff}} + \frac{1}{L_{eff}}\int_{T_{min}}^{T_{max}} V_{DD}(P_M,T)dT + \Delta I_{CMOS}(P_M,T),\ \forall T, P_M > 0$$

$$T = SRMS(T_{min},T_{max}) \Rightarrow Random_Float_Number(T_{min},T_{max})$$

$$T_{max} \geq T_{min}, \forall T_{min}, T_{max} > 0$$

Stop Pattern Generator: $P_M : I_{peak}(\forall I_{sample}(P_M,SRMS)),\ I_{averaged}(P_M,SRMS)$

$$I_{Measurement}(P_M,T) \in \{I_{peak} \quad I_{averaged}\}$$

NN_Max_Loop++ (operating loop counter)

$Sorting(P_M, I_{Measurement}(P_M)) \Rightarrow P_{selected}$ (best out of sub-optimal set via **ATE**)

if $(I_{Measurement}(P_{selected}) > \forall I_{Measurement}(P_{previous_selected}))$
{ Update VCM Database File }

if $(NN_Max_Loop < Max_Loop)$
{ *Go to* **NN_MA_Loop : New Pattern Population Generation !!!** }
else
{ Final VCM Database File Generation exit(1) }

}

Final Neural Network Maximum Approximation Plot Generation (Figure 29).

End Of NN_MA

Annex 3: Dynamic Genetic Algorithm Implementation Using ATE J973

Start D_GA: Circuit Initialization

Default AC/DC Specification Initialization.
DP Dummy Pattern : Vector Memory Initialization

$$INPUT: \{N \quad Vector_Cycles \quad DP \quad Loop1 \quad Loop2 \quad I_{Max_REF}\}$$

Check if Input valid?
else Input Error! exit(1).

For $POP = 0, 1, 2, 3, \ldots\ldots\ldots, Loop2+1$ do :
{

$$Random_Pattern_Generation \Rightarrow P \in (p_1, p_2, \ldots, p_N)$$
$$\forall vector_cycle, N > 0$$
$$P_N^{POP} = (p_1, p_2, \ldots, p_N) \text{ Initial Pattern Population}$$

For P1 = 0, 1, 2, 3, ..., $N+1$ do :
{

$$Vector_Code_Matrix(P_N(Vector_Cycles)) \Rightarrow \begin{bmatrix} P_0(Vector_Cycles) \\ P_1(Vector_Cycles) \\ \vdots \\ P_N(Vector_Cycles) \end{bmatrix}$$

$$P_N(Vector_Cycles) \in P_N(vector_encode(\forall signal_bus), Vector_Cycles))$$

$$Pattern_Generator(Vector_Memory(P_N))$$
$$\Rightarrow Pattern_Controller(Vector_Memory(P_N)) \quad N > 0 \text{ (Pattern Executor)}$$

Start Pattern Generator: $P_N(T_{\min}, T_{\max}) \Rightarrow Dynamic_Pattern$
Start Current Measurement & Calculation:

$$I_{Measurement}(P_N, T) = \frac{V_{DD}(P_N, T)}{R_{eff}} + \frac{1}{L_{eff}} \int_{T_{\min}}^{T_{\max}} V_{DD}(P_N, T)dT + \Delta I_{CMOS}(P_N, T), \ \forall T, P_N > 0$$
$$T = SRMS(T_{\min}, T_{\max}) \Rightarrow Random_Float_Number(T_{\min}, T_{\max})$$
$$T_{\max} \geq T_{\min}, \ \forall T_{\min}, T_{\max} > 0$$

Stop Pattern Generator: $P_N : I_{peak}(\forall I_{sample}(P_N, SRMS)), \ I_{averaged}(P_N, SRMS)$
$$Fixness(P_N) = I_{Measurement}(P_N, T) \in \{I_{peak} \quad I_{averaged}\}$$

}

$$Averaged_Fixness(Fixness(P_N^{POP})) = \frac{\sum_{N=0}^{N} I_{Measurement}(P_N)}{N}, N, P_N > 0$$

$if\ (Averaged_Fixness(I_{Measurement}(P_N^{POP})) > I_{Max_REF})$
{ Final VCM Generation (Database 1) exit(1) }

For P2 = 0, 1, 2, 3, ..., *Loop1*+1 do :
{

$$Sorting(I_{Measurement}(P_N) \in \{I_{\min}(P_{N_{\min}}) \quad ..... \quad I_{\max}(P_{N_{\max}})\}) \Rightarrow Parent(I_{Measurement}(P_N))$$
$$N \in \{N_{\min} \quad N_{\max}\} \qquad N \in \{N_{\min} = N_{\max} - (N_{\max} \times B) \quad N_{\max}\}$$

$$CrossOver(P_N(C_1, C_2), P_{N+1}(C_3, C_4)) \Rightarrow Upper_CrossOver(P_N(C_3, C_2), P_{N+1}(C_1, C_4))$$
$$\Rightarrow Lower_CrossOver(P_N(C_1, C_4), P_{N+1}(C_3, C_2))$$
$$\Rightarrow Stripe_CrossOver(P_N(C_4, C_3), P_{N+1}(C_2, C_1))$$

$$Recombination(P_N, P_{N+1}) \Rightarrow I_{maximum}(P_N, P_{N+1}) \Rightarrow I_{Best}(P_M),\ N, M, P_N > 0$$

$$Mutation(P_M(C_1, C_2, C_3, C_4 .... C_y)) \Rightarrow P_M(C_1 + R_1, C_2 + R_2, ... C_y + R_y)$$
$$R_y \in \{1 \quad 0 \quad -1\},\ M, P_M, y > 0$$

For P3 = 0, 1, 2, 3, ..., *M*+1 do :
{
$Pattern_Generator(Vector_Memory(P_M))$
$\Rightarrow Pattern_Controller(Vector_Memory(P_M))\ M > 0$ *(Pattern Executor)*

Start Pattern Generator: $P_M(T_{\min}, T_{\max}) \Rightarrow Dynamic_Pattern$
Start Current Measurement & Calculation:

$$I_{Measurement}(P_M, T) = \frac{V_{DD}(P_M, T)}{R_{eff}} + \frac{1}{L_{eff}} \int_{T_{\min}}^{T_{\max}} V_{DD}(P_M, T) dT + \Delta I_{CMOS}(P_M, T),\ \forall T, P_M > 0$$
$$T = SRMS(T_{\min}, T_{\max}) \Rightarrow Random_Float_Number(T_{\min}, T_{\max})$$
$$T_{\max} \geq T_{\min},\ \forall T_{\min}, T_{\max} > 0$$

Stop Pattern Generator: $P_M : I_{peak}(\forall I_{sample}(P_M, SRMS)),\ I_{averaged}(P_M, SRMS)$
$$Fixness(P_M) = I_{Measurement}(P_M, T) \in \{I_{peak} \quad I_{averaged}\}$$

}

$$Averaged_Fixness(Fixness(P_M^{POP})) = \frac{\sum_{M=0}^{M} I_{Measurement}(P_M)}{M}, M, P_M > 0$$

$if\ (Averaged_Fixness(I_{Measurement}(P_M^{POP})) > I_{Max_REF})$
{ Worst Case Pattern Found : Final VCM Generation (Database 1) exit(1) }

} *End Of Loop 1*
} *End of Loop 2*

Update So Far Worst Case Pattern Found : Final VCM Generation (Database 1)
End Of D_GA

Annex 4: Combined Neural Network Pattern Classification and Dynamic Genetic Algorithm Implementation Using ATE J973

Start NN_GA: Circuit Initialization

Default AC/DC Specification Initialization.
DP Dummy Pattern : Vector Memory Initialization

$$INPUT: \{N \quad Vector_cycle \quad DP \quad Max_Loop \quad Loop1 \quad I_{Max_REF} \quad File_Name\}$$

Check if Input valid?
else Input Error! exit(1).

$$NN_Learning_File(File_Name) \in \{w_{ij} \quad \varepsilon \quad G \quad X^k\}$$

NN_GA_Loop:
{

$$Random_Pattern_Generation \Rightarrow P \in (p_1, p_2, ..., p_N) : \text{Random Pattern Population}$$
$$\forall vector_cycle, N > 0$$

$$X^k(P_N, I_{Measurement}(P_N)) \in \{X_1^k \quad X_2^k \quad ..... \quad X_i^k\}$$ **(Neural Network Evaluation Set)**

$$X_j = \sum_i W_{ji} a_i$$

$$Y_N(X) = \frac{1}{1 + e^{-X \times G}}$$

$if(\forall Y_N(X) < Y_{sub_optimal_set})$
{ **Go to NN_MA_Loop : New Pattern Population Generation !!!** }
else
{ $Sorting(P_N, Y_N(X)) \Rightarrow P_M$ (sub-optimal set based on **neural network**) }
$P_M \Rightarrow P_N$ (replacing the old population with new sub-optimal set)

For P1 = 0, 1, 2, 3, ..., *Loop1*+1 do : (Start **genetic algorithm** to further improve sub-optimal set)
{

$$Sorting(I_{Measurement}(P_N) \in \{I_{\min}(P_{N_{\min}}) \quad ..... \quad I_{\max}(P_{N_{\max}})\}) \Rightarrow Parent(I_{Measurement}(P_N))$$
$$N \in \{N_{\min} \quad N_{\max}\} \qquad N \in \{N_{\min} = N_{\max} - (N_{\max} \times B) \quad N_{\max}\}$$

$$CrossOver(P_N(C_1,C_2),P_{N+1}(C_3,C_4)) \Rightarrow Upper_CrossOver(P_N(C_3,C_2),P_{N+1}(C_1,C_4))$$
$$\Rightarrow Lower_CrossOver(P_N(C_1,C_4),P_{N+1}(C_3,C_2))$$
$$\Rightarrow Stripe_CrossOver(P_N(C_4,C_3),P_{N+1}(C_2,C_1))$$

$$Recombination(P_N,P_{N+1}) \Rightarrow I_{maximum}(P_N,P_{N+1}) \Rightarrow I_{Best}(P_M),\ N,M,P_N>0$$

$$Mutation(P_M(C_1,C_2,C_3,C_4....C_y)) \Rightarrow P_M(C_1+R_1,C_2+R_2,.....C_y+R_y)$$
$$R_y \in \{1 \quad 0 \quad -1\},\ M,P_M,y>0$$

For P3 = 0, 1, 2, 3, ..., M+1 do :

{

$Pattern_Generator(Vector_Memory(P_M))$
$\Rightarrow Pattern_Controller(Vector_Memory(P_M))\ M>0$ *(Pattern Executor)*

Start Pattern Generator: $P_M(T_{min},T_{max}) \Rightarrow Dynamic_Pattern$
Start Current Measurement & Calculation:

$$I_{Measurement}(P_M,T) = \frac{V_{DD}(P_M,T)}{R_{eff}} + \frac{1}{L_{eff}}\int_{T_{min}}^{T_{max}} V_{DD}(P_M,T)dT + \Delta I_{CMOS}(P_M,T),\ \forall T,P_M>0$$
$$T = SRMS(T_{min},T_{max}) \Rightarrow Random_Float_Number(T_{min},T_{max})$$
$$T_{max} \geq T_{min},\ \forall T_{min},T_{max}>0$$

Stop Pattern Generator: $P_M : I_{peak}(\forall I_{sample}(P_M,SRMS)),\ I_{averaged}(P_M,SRMS)$
$$Fixness(P_M) = I_{Measurement}(P_M,T) \in \{I_{peak} \quad I_{averaged}\}$$

}

$$Averaged_Fixness(Fixness(P_M^{POP})) = \frac{\sum_{M=0}^{M} I_{Measurement}(P_M)}{M},\ M,P_M>0$$

if (Averaged_Fixness($I_{Measurement}(P_M^{POP})$) > I_{Max_REF})
{ Worst Case Pattern Found : Final VCM Generation (Database 1) exit(1) }

} *End Of Loop 1*

NN_Max_Loop++ (operating loop counter)

if (NN_Max_Loop < Max_Loop)
{ Go to **NN_GA_Loop : New Pattern Population Generation !!!** }
else
{ *So Far Worst Case Pattern Found : Final VCM Generation (Database 1)* exit(1) }

}

Final Neural Network + GA Plot Generation (Figure 31).

End Of NN_GA

EPO-BERLIN
19 -07- 2002

Claims:

1. A method of approximating the behaviour of an integrated circuit, the method comprising:
(a) applying a set of test patterns to a system for testing or simulating an integrated circuit;
(b) applying the set of test patterns to a neural network;
(c) comparing the outputs of the system for testing or simulating the integrated circuit and the outputs of the neural network; and
(d) adapting parameters of the neural network to approximate the behaviour of the integrated circuit on the basis of the comparison.

2. The method of claim 1, wherein the system for testing or simulating the integrated circuit is an automatic test equipment (ATE), and the set of test patterns is applied to the integrated circuit via the automatic test equipment.

3. The method of claim 2, wherein the neural network is implemented in the automatic test equipment.

3. The method of claim 1 or 2, wherein the set of test patterns is generated on a random basis.

4. The method of any preceding claim, wherein step (d) comprises:
adapting the inter-unit weights of the neural network through back-propagation.

5. The method of any preceding claim, comprising:

repeating steps (a) to (d) until the level of adaptation in step (d) falls below a predetermined value.

6. The method of claim 5, comprising:
storing data representing predetermined neural network parameters following termination of the repetition of steps (a) to (d).

7. A method of selecting test patterns, the method comprising:
(m) applying a test pattern to a neural network whose parameters have been adapted to approximate the behaviour of an integrated circuit according to the method of any preceding claim;
(n) processing the output of the neural network to determine whether predetermined criteria are met; and
(o) selecting for storage the test pattern if the predetermined criteria are met.

8. The method of claim 7, comprising:
repeating steps (m) to (o) until a predetermined number of test patterns have been stored.

9. The method of claim 7 or 8, wherein the predetermined criteria are met if the value of a predetermined parameter of a signal output by the neural network in response to the application of the test pattern exceeds a reference value.

10. The method of claim 9, comprising:
(p) applying a further set of test patterns to the integrated circuit using automatic test equipment (ATE);

(q) measuring the values of the predetermined parameter of output signals generated by the integrated circuit in response to step (p), wherein the predetermined criteria are met if the value of the predetermined parameter of the signal output by the neural network in response to the application of the test pattern exceeds the reference value and all values measured in this step (q).

11. The method of claim 10, wherein the further set of test patterns is generated on a random basis.

12. The method of any of claims 9 to 11, wherein the predetermined parameter is a dynamic current.

13. The method of any of claims 7 to 12, further comprising:
(r) generating a test pattern population consisting of a plurality of test patterns;
(s) applying each test pattern of the population to the neural network;
(t) for each test pattern, processing the output of the neural network to determine the value of a predetermined parameter;
(u) allocating each test pattern to one of a plurality of classification groups in accordance with the parameter value determined in step (t).

14. The method of claim 13, comprising:
repeating steps (r) to (u) using a new test pattern population consisting of the test patterns included in a selected one of the classification groups.

15. The method of claim 14, wherein the selected one of the classification groups consists of test patterns that approximate a set of worst case input parameters of operation of the integrated circuit.

16. The method of any of claims 7 to 15, comprising:
repeating steps (m) to (o) plural times;
applying the test patterns selected in each step (o) to a simulator for simulating the integrated circuit;
processing the output of the simulator to determine whether further predetermined criteria are met; and
selecting for storage those test patterns wich meet the further predetermined criteria.

17. The method of any of claims 7 to 16, comprising:
repeating steps (m) to (o) plural times;
applying the test patterns selected in each step (o) to the integrated circuit using automatic test equipment (ATE);
processing the output of the automatic test equipment to determine whether further predetermined criteria are met; and
selecting for storage those test patterns wich meet the further predetermined criteria.

18. The method of any preceding claim, wherein the predetermined criteria represent an approximation of a worst case mode of operation of the integrated circuit.

19. A method of simulating an integrated circuit; the method comprising:
applying test patterns that have been selected according to the method of any of claims 10 to 18 to a simulator for simulating the integrated circuit.

20. A method of testing an integrated circuit; the method comprising:
applying test patterns that have been selected according to the method of any of claims 10 to 19 to the integrated circuit using automatic test equipment (ATE).

21. A method of providing a test pattern for the simulation and/or test of the layout of an integrated circuit, the method comprising the steps of:
(A) providing a set of test patterns consisting of test patterns selected in accordance with the method of any of claims 7 to 18;
(B) applying the set of test patterns to the integrated circuit using automatic test equipment (ATE);
(C) determining the outputs of the integrated circuit;
(D) processing the outputs to determine whether predetermined test criteria are met;
(E) depending on the determination in step (D), generating a new set of test patterns on the basis of the set of test patterns provided by step (A) using a genetic algorithm.

22. The method of claim 21, comprising:
(F1) repeating steps (B) to (E) until the predetermined test criteria are met.

23. The method of claim 21, comprising:
(F2) repeating steps (B) to (E) for a predetermined number of times or until the predetermined test criteria are met.

24. The method of any of claims 21 to 23, wherein the predetermined test criteria are met if the set of test

patterns is associated with an average fitness above a predetermined value.

25. The method of any of claims 21 to 24, wherein step (E) comprises:
combining some or all of the test patterns according to the genetic algortihm to provide the new set of test patterns.

26. The method of claim 25, further comprising:
selecting test patterns out of the set of test patterns according to predetermined selection criteria; and
combining the selected test patterns according to the genetic algortihm to provide the new set of test patterns

27. The method of claim 26, comprising:
selecting a test pattern if it is associated with a fitness value greater than a reference value.

28. The method of claim 26 or 27, comprising:
(G) selecting a test pattern if it is associated with the highest fitness value of all unselected test patterns.

29. The method of claim 28, comprising:
repeating step (G) until a predetermined percentage of test patterns has been selected.

30. The method of claim 28, wherein step (E) comprises:
(H) arranging the selected test patterns in the order of associated fitness values;
(I) randomly selecting parent test patterns out of the test patterns as arranged in step (H); and

(J) combining the selected parent test patterns.

31. The method of any of claims 21 to 30, wherein the genetic algorithm includes crossing over, re-combination, and/or mutation of selected test patterns.

32. The method of any of claims 21 to 31, wherein step (A) comprises:
providing a plurality of sets of test patterns, wherein each set of test patterns is included in a test pattern population.

33. The method of claim 32, comprising:
performing the steps of any of claims 21 to 32 for each population.

34. Data processing system, adapted to perform the method of any preceding claim.

35. Computer program, for performing the method of any of claims 1 to 33 on a data processing system.

EPO-BERLIN
19 -07- 2002

Abstract

Method of processing test patterns for an integrated circuit

There is provided a method of approximating the behaviour of an integrated circuit, the method comprising: (a) applying a set of test patterns to a system for testing or simulating an integrated circuit; (b) applying the set of test patterns to a neural network; (c) comparing the outputs of the system for testing or simulating the integrated circuit and the outputs of the neural network; and (d) adapting parameters of the neural network to approximate the behaviour of the integrated circuit on the basis of the comparison. In particular, the system for testing or simulating the integrated circuit may be an automatic test equipment (ATE), and the set of test patterns is applied to the integrated circuit via the automatic test equipment. The invention is based on the idea that the dynamic behaviour of the integrated circuit device can be learnt from a set of random test patterns using a neural network. After the learning process has been completed, the ATE is able to perform test pattern classification. The ATE may thus select test patterns for subsequent simulation or testing of the integrated circuit. Preferably, the selected patterns are further optimized using a genetic algorithm.

Fig. 5

EPO-BERLIN
19 -07- 2002

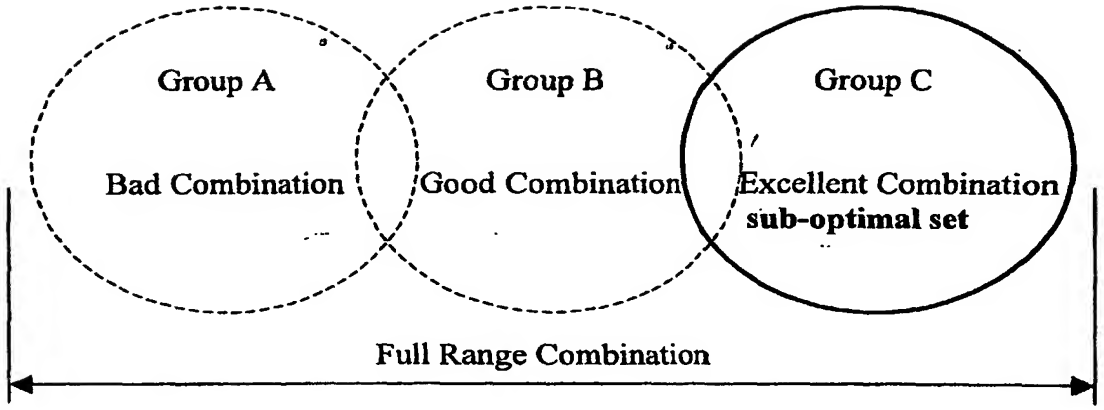

Fig. 1

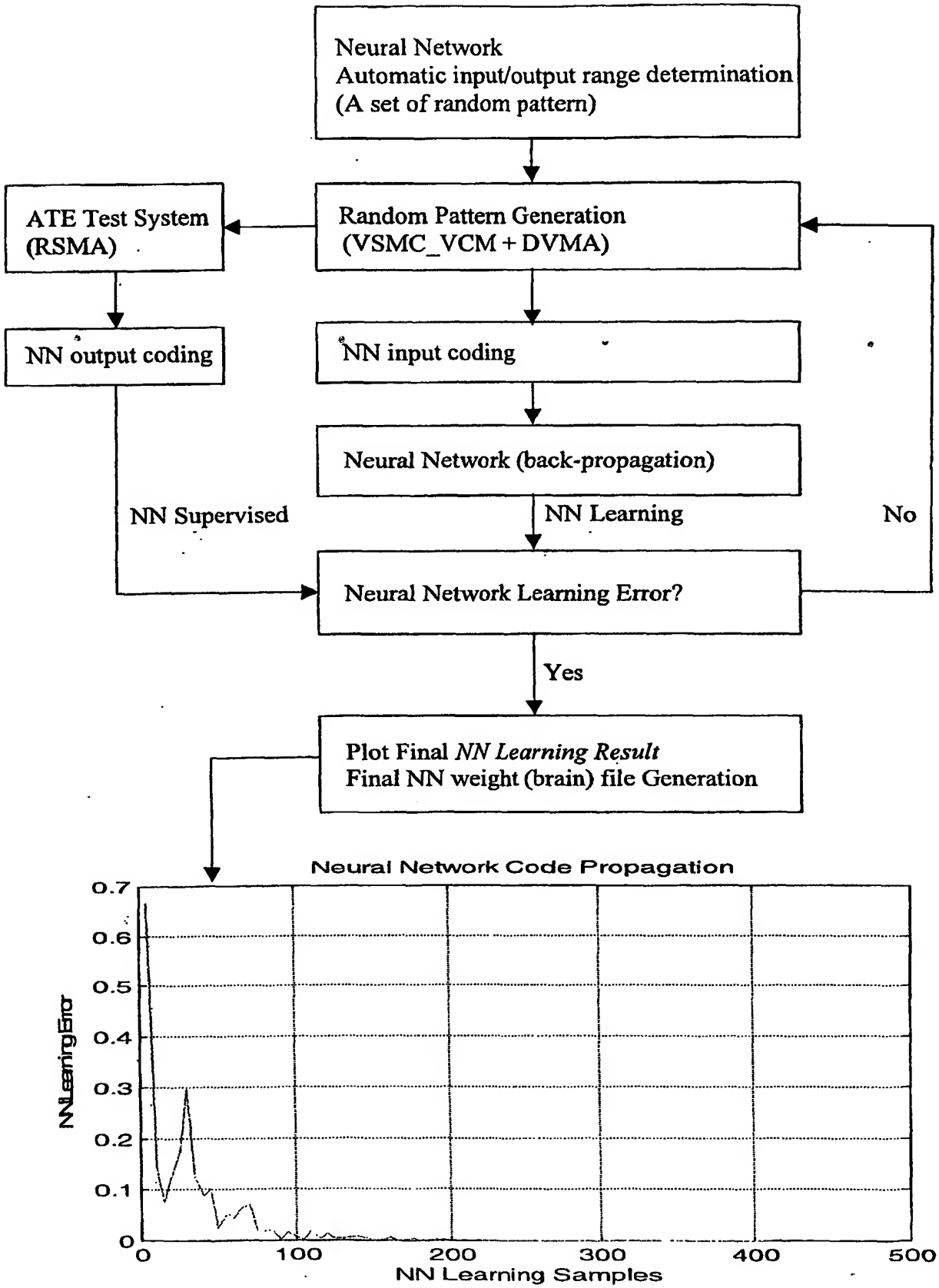
Neural Network
Automatic input/output range determination
(A set of random pattern)
ATE Test System
(RSMA)
Random Pattern Generation
(VSMC_VCM + DVMA)
NN output coding
NN input coding
Neural Network (back-propagation)
NN Supervised
NN Learning
No
Neural Network Learning Error?
Yes
Plot Final *NN Learning Result*
Final NN weight (brain) file Generation
Neural Network Code Propagation
NNLearningError
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
100
200
300
400
500
NN Learning Samples

fig. 2

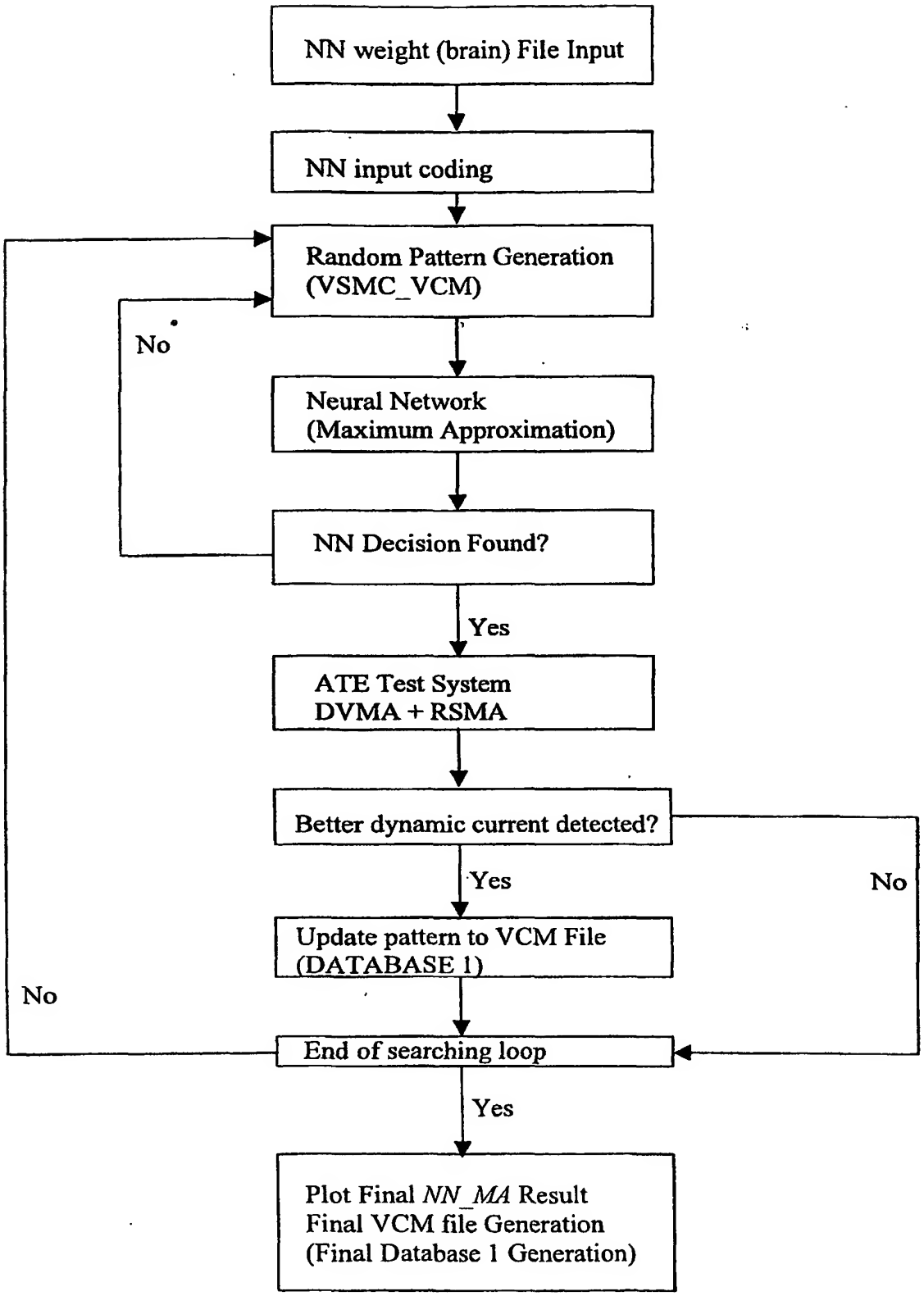
NN weight (brain) File Input
NN input coding
Random Pattern Generation (VSMC_VCM)
No
Neural Network (Maximum Approximation)
NN Decision Found?
Yes
ATE Test System DVMA + RSMA
Better dynamic current detected?
Yes
No
Update pattern to VCM File (DATABASE 1)
No
End of searching loop
Yes
Plot Final NN_MA Result
Final VCM file Generation
(Final Database 1 Generation)

fig. 3

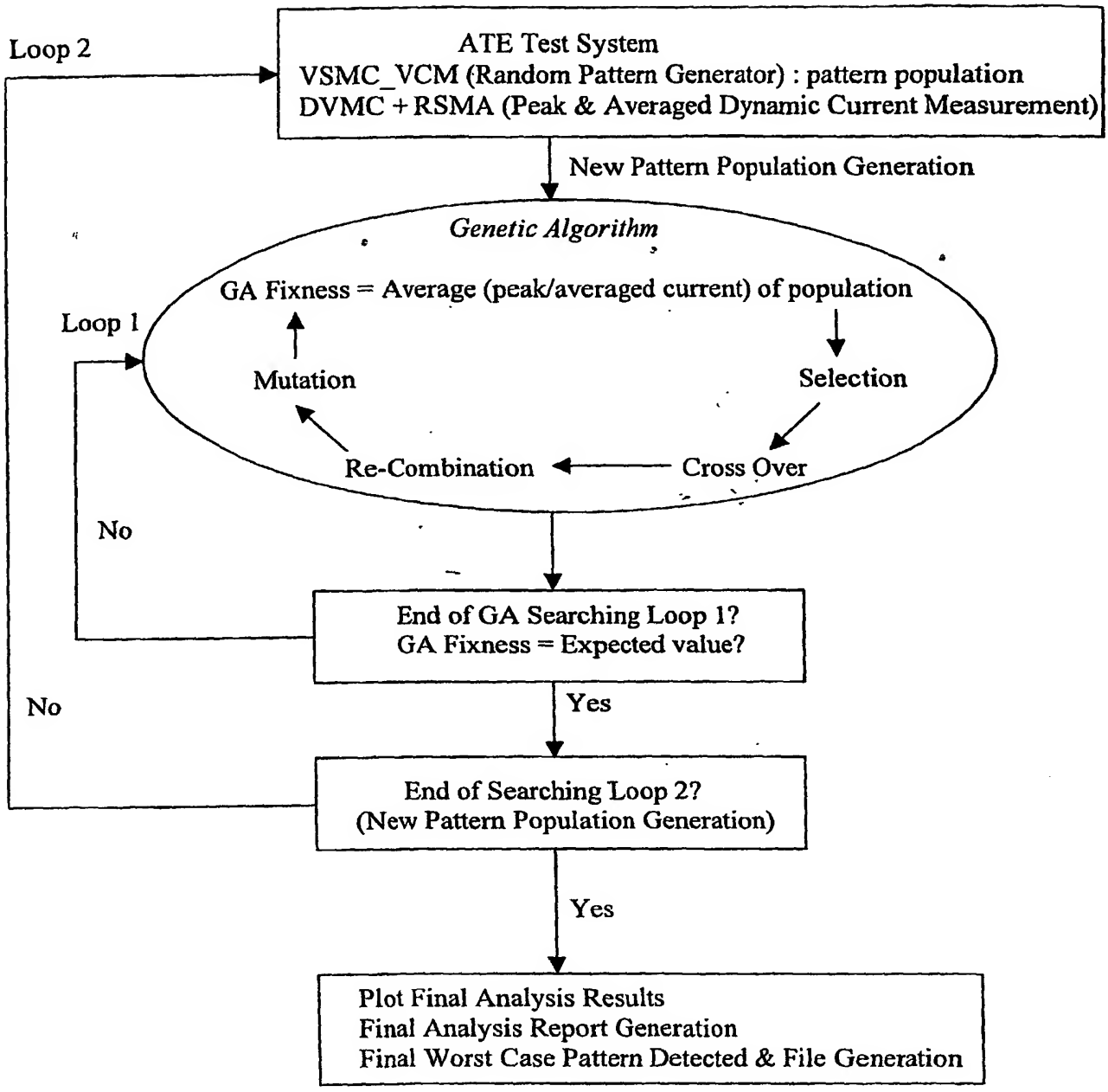
Loop 2
ATE Test System
VSMC_VCM (Random Pattern Generator) : pattern population
DVMC + RSMA (Peak & Averaged Dynamic Current Measurement)
New Pattern Population Generation
Genetic Algorithm
GA Fixness = Average (peak/averaged current) of population
Loop 1
Mutation
Selection
Re-Combination
Cross Over
No
End of GA Searching Loop 1?
GA Fixness = Expected value?
Yes
No
End of Searching Loop 2?
(New Pattern Population Generation)
Yes
Plot Final Analysis Results
Final Analysis Report Generation
Final Worst Case Pattern Detected & File Generation

Fig. 4

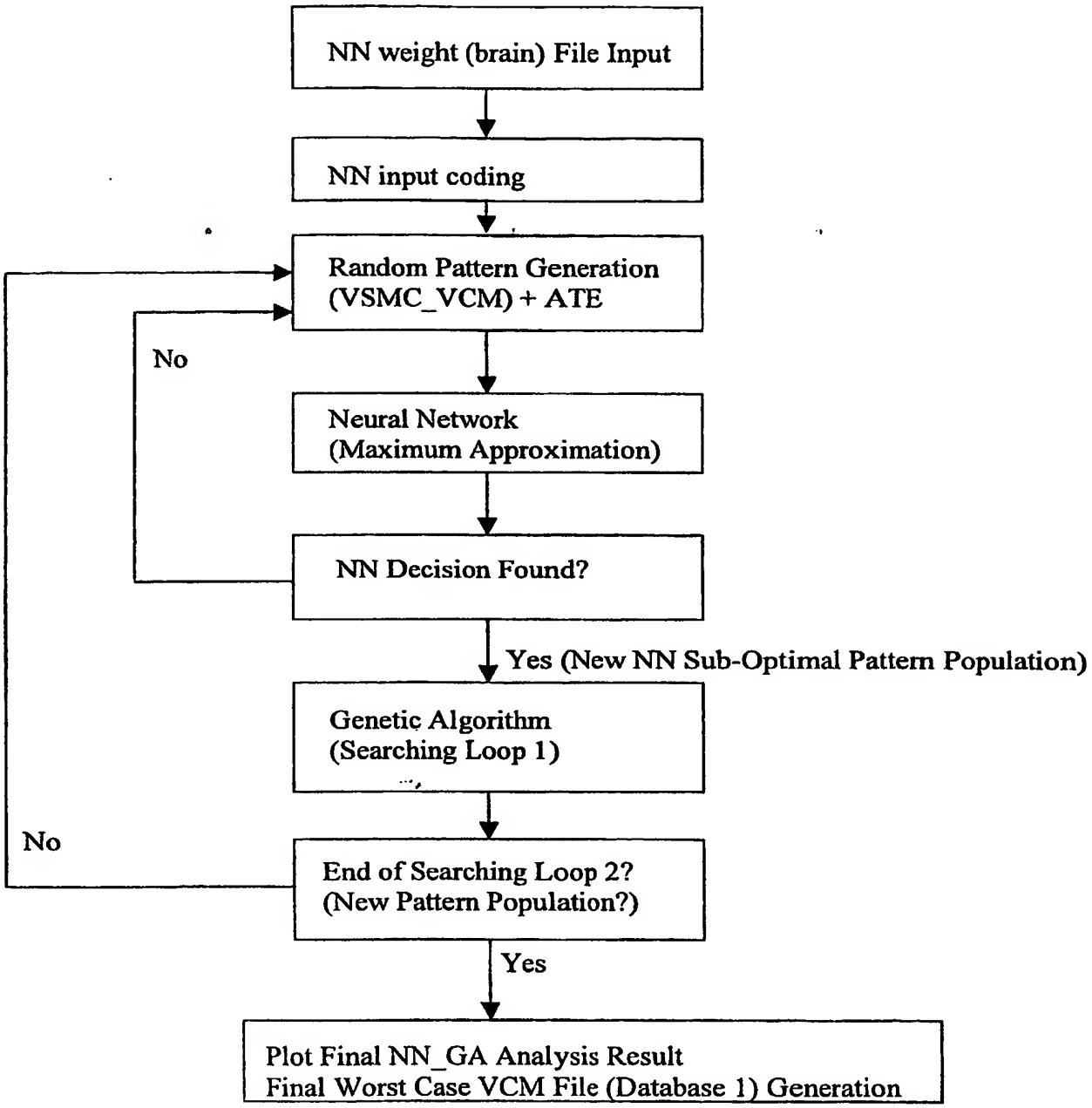
NN weight (brain) File Input
NN input coding
Random Pattern Generation
(VSMC_VCM) + ATE
No
Neural Network
(Maximum Approximation)
NN Decision Found?
Yes (New NN Sub-Optimal Pattern Population)
Genetic Algorithm
(Searching Loop 1)
No
End of Searching Loop 2?
(New Pattern Population?)
Yes
Plot Final NN_GA Analysis Result
Final Worst Case VCM File (Database 1) Generation

Fig. 5